

Oefenzitting 7 - Functies IV (Activatie Records)

Oefening 1

Vertaal het volgende C programma.

```
struct elem {
    int type;
    int aantal;
};
struct elem stock[10];

int isType (register int soort, struct elem * obj) {
    return (obj->type == soort); // 1 indien gelijk, anders 0
}

void increment (struct elem rij[], int * soort,
               register int start) {
    struct elem tmp;
    tmp = rij[start];
    if (isType(*soort, &tmp)) {
        rij[start].aantal++;
        increment (rij, soort, start+1);
        printint(start);
    }
}

struct elem maakElem (register int ras) {
    struct elem tmp;
    tmp.type = ras;
    tmp.aantal = 1;
    return (tmp);
}

main () {
    int ras;
    register int i;
    register struct elem * ptr;
    for (i = 0; i < 10; i++) {
        stock[i] = maakElem(i);
    }
    ras = 5;
    increment (stock, &ras, 0);
    ras = 7;
    increment (stock, &ras, 5);
    for (ptr = stock; ptr < &stock[10]; ptr++) {
        printint(ptr->type, ptr->aantal);
    }
}
```

```

    }
}

```

Oefening 2

In de vorige opgave, vervang `maakElem` door `maakElem2` en vergelijk beide:

```

struct elem maakElem2 (register int ras) {
    struct elem * nieuw;
    nieuw = alloc(2);
    nieuw->type = ras;
    nieuw->aantal = 1;
    return (*nieuw);
}

```

Oefening 3

In de vorige opgave, vervang `maakElem2` door `maakElem3` en pas het hoofdprogramma als volgt aan:

```

struct elem * maakElem3 (register int ras) {
    struct elem * nieuw;
    nieuw = alloc(2);
    nieuw->type = ras;
    nieuw->aantal = 1;
    return (nieuw);
}

main () {
    int ras;
    register int i;
    register struct elem * ptr;
    for (i = 0; i < 10; i++) {
        ptr = maakElem3(i);
        stock[i] = *ptr;
    }
    // rest blijft hetzelfde
}

```

Kan je `maakElem` ook aanpassen zodat een wijzer wordt teruggegeven naar het record i.p.v. het record zelf?

Oefening 4

Laatste wijziging: pas de declaratie van `stock` aan en wijzig `increment`.

```
struct elem * stock[10];

void increment2 (struct elem * rij[], int * soort,
                register int start) {
    struct elem * tmp;
    tmp = rij[start];
    if (isType(*soort, tmp)) {
        rij[start]->aantal++;
        increment (rij, soort, start+1);
        printint(start);
    }
}

main () {
    int ras;
    register int i;
    register struct elem * ptr;
    for (i = 0; i < 10; i++) {
        stock[i] = maakElem3(i);
    }
    // rest blijft hetzelfde, maar vervang increment door increment2
    for (ptr = stock; ptr < &stock[10]; ptr++) {
        printint((*ptr)->type, (*ptr)->aantal);
    }
}
```

Hoe vergelijk je deze oplossing met de voorgaande? Hoe zit het met het geheugengebruik?